FP31VecMath Logic

Multiplication

Start with an array of N, Vec8ui values. (Each value is a FP31Val (base31): an un-signed 32 bit value, where the most significant bit is reserved for over-flow detection)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 8, 32-bit lanes | | | | | | | |
| Limb | A | B | C | D | E | F | G | H |
| 0 |  |  |  |  |  |  |  |  |
| 1, 2, 3… |  |  |  |  |  |  |  |  |
| N - 1 |  |  |  |  |  |  |  |  |

Take the two’s compliment and unpack into 16, 32-bit values.

1. Get the signs of each value by creating a Vec8ui where each value is true (0xFFFFFFFF) if bit 30 of Limb at index N-2 is 0, or false (0x00000000) if bit 30 is set.)
   1. \_mm256\_and\_si256 (Limb0, 0x40000000) // returns non-zero value if the limb’s sign bit is set.
   2. \_mm256\_cmpeq\_epi32 (result, 0x00000000) // if sign bit was set, returns false: (0x00000000) otherwise returns true: (0xFFFFFFFF)
   3. \_mm256\_movemask\_epi8 (treating the result as a Vec32c, create a 32-bit integer from the most significant bit of each 8-bit value.
2. If all 8 values are positive (i.e., the result is -1), simply unpack by
   1. For each Limb,
      1. Create a Vec4u from \_mm256\_permutevar8x32\_epi32(limb, (0x00112233) to hold the lower 4 values.
      2. Create a Vec4u from \_mm256\_permutevar8x32\_epi32(limb, (0x44556677) to hold the upper 4 values.
3. Otherwise
   1. For each Limb
      1. ….
      2. ….
      3. ….

Now we have a pair of arrays of N, Vec4ui values; one array to hold the first four values and another to hold the second four values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 4, 64-bit lanes | | | |
| Limb | A | B | C | D |
| 0 |  |  |  |  |
| 1, 2, 3… |  |  |  |  |
| N - 1 |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 4, 64-bit lanes | | | |
| Limb | E | F | G | H |
| 0 |  |  |  |  |
| 1, 2, 3… |  |  |  |  |
| N - 1 |  |  |  |  |